(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2003－223360
（P2003－223360A）
(43)公開日　平成15年８月８日(2003.8.8)

(51)Int.Cl.'	識別記号	ＦＩ	テーマコード(参考)
Ｇ０６Ｆ　12/08	５０７	Ｇ０６Ｆ　12/08	５０７Ｆ　５Ｂ００５
	５０１		５０１Ｃ
	５５９		５５９Ｅ
			５５９Ｚ
	５６１		５６１

審査請求　未請求　請求項の数７　ＯＬ　（全 14 頁）　最終頁に続く

(21)出願番号　特願2002－19905(P2002－19905)

(22)出願日　平成14年１月29日(2002.1.29)

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　田原　康宏
東京都小平市上水本町五丁目20番１号　株式会社日立製作所半導体グループ内
(74)代理人　100080001
弁理士　筒井　大和

Ｆターム(参考)　5B005 JJ11 KK12 MM01 NN45 PP03 QQ05

(54)【発明の名称】　キャッシュメモリシステムおよびマイクロプロセッサ

(57)【要約】
【課題】　主記憶装置とキャッシュメモリとの間の不要なデータ転送を削減して、データ転送に伴う電力消費を削減し、データ転送の経路の混雑を緩和してスループットを向上させることができるキャッシュメモリ制御技術を提供する。
【解決手段】　メインＣＰＵと、ＲＯＭとＲＡＭからなる主記憶装置とが外部バスを通じて相互に接続されているキャッシュメモリシステムであって、４－ｗａｙセットアソシエイティブキャッシュからなり、各ＷａｙはＴａｇ４５、Ｖａｌｉｄビット４６、Ｄｉｒｔｙビット４７、データブロック４８を持つ。キャッシュエントリをキャッシュから追い出すときに、Ｄｉｒｔｙビット４７が１にセットされていたらデータブロック４８のデータを主記憶装置に書き込み、０にクリアされていたらデータブロック４８のデータを主記憶装置に書き込まないで捨ててよい。

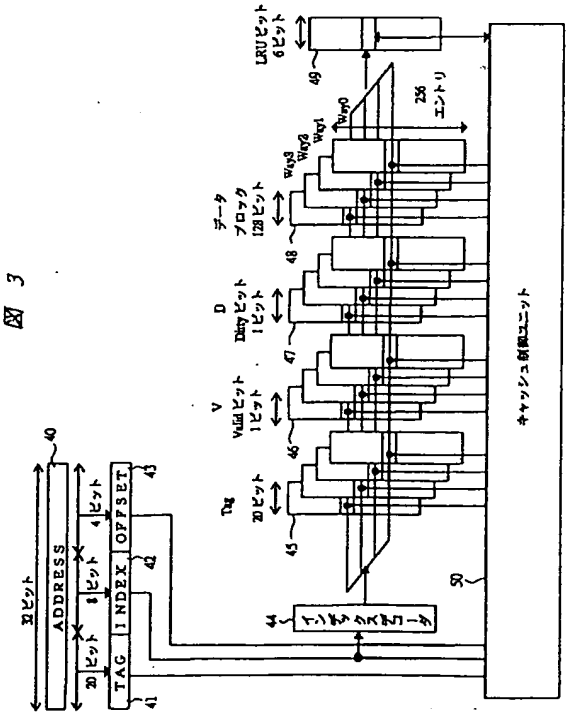


図３

【特許請求の範囲】

【請求項１】　ライトバック方式のキャッシュメモリを有し、

前記キャッシュメモリ上に確保した領域のダーティビットを操作する命令を実行可能とすることを特徴とするキャッシュメモリシステム。

【請求項２】　請求項１記載のキャッシュメモリシステムにおいて、

前記命令はメモリ解放命令であり、このメモリ解放命令のニモニックとオペランドは、

ＭＲＥＬ　Ｒｎ，ＩＭＭ

Ｒｎ：レジスタ名

ＩＭＭ：所定のビット数の即値

であることを特徴とするキャッシュメモリシステム。

【請求項３】　請求項１記載のキャッシュメモリシステムにおいて、

前記命令はダーティビットクリア命令であり、このダーティビットクリア命令のニモニックとオペランドは、

ＤＣＢＤＣ　＠Ｒｎ

Ｒｎ：はレジスタ名

であることを特徴とするキャッシュメモリシステム。

【請求項４】　請求項２または３記載のキャッシュメモリシステムにおいて、

前記Ｒｎ：レジスタ名は、スタック領域に動的に確保したメモリ領域を解放するために使用可能なレジスタであることを特徴とするキャッシュメモリシステム。

【請求項５】　請求項２または３記載のキャッシュメモリシステムにおいて、

前記Ｒｎ：レジスタ名は、ヒープ領域に動的に確保したメモリ領域を解放するために使用可能なレジスタであることを特徴とするキャッシュメモリシステム。

【請求項６】　請求項１記載のキャッシュメモリシステムにおいて、

前記命令の実行を指示するプロセッサと、主記憶装置とをさらに有し、

前記プロセッサの指示に基づいて前記主記憶装置内の特定のアドレスが指す領域を確保して所定のプログラム処理を行い、

前記確保した領域を前記キャッシュメモリ上に確保して使用した後に、前記プロセッサの指示により前記主記憶装置に確保した領域を解放するとき、前記キャッシュメモリ上に確保した領域のダーティビットをクリアするように制御することを特徴とするキャッシュメモリシステム。

【請求項７】　ライトバック方式のキャッシュメモリを有し、

前記キャッシュメモリ上に確保した領域のダーティビットを操作する命令を実行可能とすることを特徴とするマイクロプロセッサ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、キャッシュメモリ制御技術に関し、たとえばマイクロプロセッサ、マイクロコンピュータ、マイクロコントローラのキャッシュメモリなどを有するキャッシュメモリシステムにおいて、特にダイナミックに確保したメモリ領域を解放するときのキャッシュメモリ制御方式に適用して有効な技術に関する。

【０００２】

【従来の技術】本発明者が検討したところによれば、キャッシュメモリ制御技術については、たとえば特開平１１－３３８７７２号、特開平４－２６８６３８号、特開平４－２６４６４１号、特開平４－１８８３２６号、特開平６－２８２５３号、特開平９－２３１１３３号の各公報に記載される技術などが挙げられる。

【０００３】前記公報の技術においては、主記憶装置にダイナミックに確保した領域をキャッシュメモリのエントリとして読み込んだ状態で、主記憶装置の当該領域を解放したときに、当該キャッシュエントリのＶａｌｉｄビットをクリアすることにより当該キャッシュエントリを無効にしている。

【０００４】

【発明が解決しようとする課題】ところで、前記のようなキャッシュメモリ制御技術について、本発明者が検討した結果、以下のようなことが明らかとなった。以下において、本発明者が本発明の前提として検討した技術を、図１０～図１２を用いて説明する。図１０はスタックの使用方法、図１１はキャッシュの状態、図１２はＩＮＤＥＸが０ｘＡ８のキャッシュエントリ、をそれぞれ示す説明図である。

【０００５】［スタックの一般的使い方の説明：図１０］使用前のスタックポインタＳＰは、ＳＰ＝０ｘＦＥＤＣＢＡ９０という値を持つと仮定する。０ｘは、次に続く文字列が１６進数であることを示す接頭辞である。スタックに０ｘ１０バイトの領域を確保するときは、

ＳＰ←ＳＰ－０ｘ１０

を実行する。すると、ＳＰ＝０ｘＦＥＤＣＢＡ８０となる。

【０００６】ユーザ旧ＳＰ＝０ｘＦＥＤＣＢＡ９０と、現ＳＰ＝０ｘＦＥＤＣＢＡ８０との間にある０ｘ１０バイトのメモリ領域が使用可能となる。たとえば、

（ＳＰ＋０）番地にデータ０ｘ０１２３４５６７、

（ＳＰ＋４）番地にデータ０ｘ８９ＡＢＣＤＥＦ、

（ＳＰ＋８）番地にデータ０ｘ０１２３４５６７、

（ＳＰ＋１２）番地にデータ０ｘ８９ＡＢＣＤＥＦ、

を書き込む。これらのデータが不要になり、もはやこれらのデータのためにメモリ領域を確保しておく必要がなくなった時に、

ＳＰ←ＳＰ＋０ｘ１０

を実行する。すると、ＳＰは旧ＳＰの値に戻り、確保さ

れた０ｘ１０バイトのメモリ領域は解放される。

【０００７】スタック用メモリ領域のうち、ＳＰより若いアドレスにあるメモリ領域は未使用で解放されているメモリ領域である。スタック用メモリ領域のうち、ＳＰよりアドレスが大きいメモリ領域は既に確保されているメモリ領域である。

【０００８】［ライト時のキャッシュ動作：図１１］図１１は、（ＳＰ＋４）番地、つまり０ｘＦＥＤＣＢＡ８４番地にデータ０ｘＦＦＥＥＤＤＣＣを書き込んだ後のキャッシュの状態を示している。

【０００９】書き込むアドレス０ｘＦＥＤＣＢＡ８４がＡＤＤＲＥＳＳ６０として扱われ、ＡＤＤＲＥＳＳのＭＳＢ側から２０ビットはＴＡＧ６１、次の８ビットはＩＮＤＥＸ６２、最後の４ビットはＯＦＦＳＥＴ６３に分解して扱う。Ｗａｙ　Ｊ６４のＩＮＤＥＸ＝０ｘＡ８のエントリが本書き込み前にＩｎｖａｌｉｄ状態（ＶａｌｉｄビットＶ＝０）となっていて、Ｗａｙ　Ｊが本書き込みのために選ばれたと仮定する。本エントリでＶ＝０のとき、Ｔａｇ、Ｄｉｒｔｙビット（Ｄビット）、データブロックには不定値が入っているが、本書き込みによりＴａｇに０ｘＦＥＤＣＢ、Ｖビットに１、Ｄビットに１、データブロックのオフセットがＯＦＦＳＥＴ＝４の位置から４バイトにデータ０ｘＦＦＥＥＤＤＣＣが書かれる。

【００１０】本説明では、ライトバック方式のキャッシュメモリを仮定しているため、主記憶装置とキャッシュエントリのデータブロックとのコヒーレンシがとれない場合に、Ｄビットを１にセットする。本データブロックの残りの部分にはライトアロケート機能により、主記憶装置の対応するアドレスからデータをリードする。このときの状態が図１１である。

【００１１】［本発明の前提技術の具体的な説明：図１２］本発明の前提技術では、

ＳＰ←ＳＰ＋０ｘ１０

で、スタックの０ｘ１０バイトのメモリ領域を開放すると同時に、解放したメモリ領域が割り付けられているキャッシュエントリのＶａｌｉｄビット（Ｖビット）を０にクリアしてＩｎｖａｌｉｄにしている。仮に、Ｖビットをクリアしない場合、本キャッシュエントリがＬＲＵアルゴリズムによってキャッシュの外に追い出されるときに、Ｄビットが１ならばデータブロックの内容が主記憶装置に書き戻される。しかし、本来このデータブロックの内容は既に解放された領域のデータなので、主記憶装置に書き戻しても再度利用されることはなく、書き戻しは無駄である。従って、書き戻しが発生しないようにＶビットをクリアすることにより、無用な書き戻しが発生しないようにしていた。当該キャッシュエントリをＩｎｖａｌｉｄにした状態を図１２（ａ）に示す。

【００１２】［本発明の前提技術の課題］前述した本発明の前提技術の課題は、ライトアロケート方式のキャッシュメモリにおいて、一度解放した領域と同一のアドレス範囲を再度確保した場合に露呈する。本発明の前提技術では、解放したときにＶａｌｉｄビットをクリアするので、当該キャッシュエントリが無効になり、タグやデータを失う。同じアドレス範囲を再度確保して最初に書き込んだときに、タグが失われているので、新たにキャッシュエントリを割り付ける必要がある。ライトアロケート方式に従うと、書き込んだデータを除くキャッシュエントリのデータを主記憶装置からキャッシュエントリに読み込む。このデータ転送により電力を消費したり、データ転送の経路を混雑させて性能が劣化することが、課題として考えられる。この課題を克服することが本発明の目的である。

【００１３】たとえば、０ｘＦＥＤＣＢＡ８０から始まる１６バイトのスタック領域を解放したときに、本発明の前提技術により図１２（ａ）のように当該キャッシュエントリをＩｎｖａｌｉｄにしたと仮定する。次に、０ｘＦＥＤＣＢＡ８４番地に０ｘＢＢＢＢＡＡＡＡという値を書き込み、同じＷａｙにアロケートされたと仮定すると、図１２（ｂ）の状態になる。このとき、当該データブロックのライトした４バイトデータを除く部分は主記憶装置から読み込まれる。この読み込みにより消費される電力を削減することと、この読み込みによるデータを転送する経路の混雑をなくし、スループットを向上することが本発明の目的である。

【００１４】すなわち、本発明の目的は、主記憶装置とキャッシュメモリとの間の不要なデータ転送を削減して、データ転送に伴う電力消費を削減し、データ転送の経路の混雑を緩和してスループットを向上させることができるキャッシュメモリ制御技術を提供することにある。

【００１５】本発明の前記ならびにその他の目的と新規な特徴は、本明細書の記述および添付図面から明らかになるであろう。

【００１６】

【課題を解決するための手段】本願において開示される発明のうち、代表的なものの概要を簡単に説明すれば、次のとおりである。

【００１７】本発明によるキャッシュメモリシステムは、ライトバック方式のキャッシュメモリを有し、このキャッシュメモリ上に確保した領域のダーティビットを操作する命令を実行可能とするものである。特に、前記命令は、メモリ解放命令ＭＲＥＬ　Ｒｎ，ＩＭＭ、あるいはダーティビットクリア命令ＤＣＢＤＣ　@Ｒｎ、であり、スタック領域、あるいはヒープ領域に動的に確保したメモリ領域を解放するために使用可能なレジスタに適用するものである。

【００１８】さらに、前記キャッシュメモリシステムにおいて、命令の実行を指示するプロセッサと、主記憶装置とを有し、プロセッサの指示に基づいて主記憶装置内

の特定のアドレスが指す領域を確保して所定のプログラム処理を行い、この確保した領域をキャッシュメモリ上に確保して使用した後に、プロセッサの指示により主記憶装置に確保した領域を解放するとき、キャッシュメモリ上に確保した領域のダーティビットをクリアするように制御するものである。

【００１９】すなわち、本発明は、本発明の前提技術において、主記憶装置の解放した領域に対応するキャッシュエントリのＶａｌｉｄビットをクリアしていたのに対して、本発明では当該キャッシュエントリのＤｉｒｔｙビットをクリアする点を特徴とするものである。

【００２０】これにより、Ｄｉｒｔｙビットをクリアしたキャッシュエントリがキャッシュに残っている場合に、同一のアドレス範囲にある主記憶装置の領域を新たに確保し、この領域に最初にデータを書き込んだ時に本発明の効果が現れる。既に当該アドレス範囲に対応するキャッシュエントリがキャッシュに存在するため、ライトアロケートする必要がなく、データ転送が発生しない。よって、データ転送のための電力消費を発生せず、データ転送の経路を混雑させて性能を劣化させることもない。

【００２１】また、ＤｉｒｔｙビットをクリアしたキャッシュエントリがＬＲＵアルゴリズムなどによりキャッシュから追い出される場合においても、Ｄｉｒｔｙビットがクリアされているために、キャッシュエントリのデータを主記憶装置に書き戻す必要がなく、書き戻しのためのデータ転送が発生しない。よって、データ転送のための電力消費を発生せず、データ転送の経路を混雑させて性能を劣化させることもない。

【００２２】たとえば、０ｘＦＥＤＣＢＡ８０から始まる１６バイトのスタック領域が、前述した図１１のようにＷａｙ　ＪのＩＮＤＥＸ＝０ｘＡ８のキャッシュエントリに割り付けられている状態で、本領域を解放したときに、本発明では図１２（ｃ）のように当該キャッシュエントリのＤｉｒｔｙビットを０にクリアする。このキャッシュエントリがキャッシュの外に追い出されないうちに０ｘＦＥＤＣＢＡ８４番地に０ｘＢＢＢＢＡＡＡＡという値を書き込むと、図１２（ｄ）の状態になる。このとき、当該データブロックのライトしたデータを除く部分は主記憶装置から読み込まれることはなく、データブロックに元々書かれていた値である。従って、本発明の前提技術ではライトアロケート方式により主記憶装置からの読み込みが発生していたが、本方式ではこの読み込みが発生しない。この読み込みにより消費される電力を削減され、この読み込みによるデータを転送する経路の混雑は発生せず、スループットを向上することができる。

【００２３】また、図１２（ｃ）の状態のキャッシュエントリがＬＲＵアルゴリズムによりキャッシュの外に追い出されるとき、Ｄｉｒｔｙビットが０にクリアされて

いるため、書き戻しのためのデータ転送の電力消費を発生せず、データ転送の経路を混雑させて性能を劣化させることもない。

【００２４】

【発明の実施の形態】以下、本発明の実施の形態を図面に基づいて詳細に説明する。

【００２５】まず、図１により、本発明を応用した本発明の一実施の形態の携帯電話システムの構成の一例を説明する。図１は本実施の形態の携帯電話システムを示す説明図であり、（ａ）は平面図、（ｂ）は機能ブロック図をそれぞれ示す。

【００２６】本実施の形態の携帯電話システムは、無線信号を送受信する無線部１０１、送受信信号を変調／復調処理するベースバンド回路１０２、信号をフィルタ処理するＤＳＰ１０３、アナログ／デジタル変換するＡ／Ｄ変換器１０４、信号増幅するＡＦ回路１０５、音声を出力するスピーカ１０６、音声を入力するマイク１０７からなる電話機能部と、表示信号を演算処理する操作部・ＣＰＵ１０８、表示用のＬＣＤ１０９、ＬＣＤ１０９を駆動するＬＣＤドライバ１１０、キー入力するキー入力部１１１からなる表示機能部と、全体の演算処理を司るメインＣＰＵ１１２、主記憶装置のＲＯＭ１１３、ＲＡＭ１１４、フラッシュメモリ１１５からなる制御機能部などから構成されている。

【００２７】電話機能部における受信時は、無線信号を無線部１０１で受信すると、この無線信号がベースバンド回路１０２により復調処理され、さらにＤＳＰ１０３によりフィルタ処理された後に、Ａ／Ｄ変換器１０４によりアナログ信号からデジタル信号に変換される。そして、デジタル信号は、ＡＦ回路１０５により増幅され、スピーカ１０６を通じて音声信号として出力される。

【００２８】さらに、送信時は、マイク１０７から音声を入力すると、音声信号がＡＦ回路１０５により増幅され、さらにＡ／Ｄ変換器１０４によりデジタル信号からアナログ信号に変換される。そして、ＤＳＰ１０３によりフィルタ処理され、ベースバンド回路１０２により変調処理された後に、無線部１０１を通じて無線信号として送信される。

【００２９】また、表示機能部においては、電話の情報や付加的に設けられた電子メールなどの各種情報が操作部・ＣＰＵ１０８により演算処理され、これらの各種情報がＬＣＤドライバ１１０を介してＬＣＤ１０９に表示される。また、キー入力部１１１からの入力により、各種機能の選択や、電子メールの文字入力などを行うことができる。

【００３０】また、制御機能部においては、携帯電話システムの全体の演算処理がメインＣＰＵ１１２で実行され、このメインＣＰＵ１１２による各種演算処理は、たとえばＲＯＭ１１３やフラッシュメモリ１１５に記憶されている各種プログラムに基づいて行われ、これらの各

種演算処理のデータは随時、たとえばＲＡＭ１１４に格納される。この制御機能部のメインＣＰＵ１１２、ＲＯＭ１１３、ＲＡＭ１１４からなる部分についての詳細は後述する。

【００３１】次に、図２により、前記図１のメインＣＰＵとＲＯＭとＲＡＭからなる部分のキャッシュメモリシステムの構成の一例を説明する。図２はメインＣＰＵとＲＯＭとＲＡＭからなる部分のキャッシュメモリシステムを示すブロック図である。

【００３２】図２において、キャッシュメモリシステムは、プロセッサであるメインＣＰＵ１と、ＲＯＭとＲＡＭからなる主記憶装置２とは外部バス３を通じて相互に接続されている。なお、図２におけるメインＣＰＵ１は図１のメインＣＰＵ１１２に対応し、また図２における主記憶装置２のＲＯＭとＲＡＭは図１のＲＯＭ１１３とＲＡＭ１１４にそれぞれ対応する。

【００３３】メインＣＰＵ１には、データキャッシュ１０、命令キャッシュ１１、ライトバッファ１２、制御ユニット１３、命令バッファ１４、命令デコードユニット１５、レジスタファイル１６、スタックポインタ６、プログラムカウンタ１７、演算回路１８、メモリデータアクセスユニット１９、ライトバックユニット２０、バスユニット２１が設けられ、これらの各ユニットは内部アドレスバス２２、内部データバス２３に任意に接続されている。また、このメインＣＰＵ１のバスユニット２１は、外部アドレスバス２４、外部データバス２５、外部制御信号線２６を通じて、外部バス３、主記憶装置２に接続されている。

【００３４】主記憶装置２には、ＲＯＭ４、ＲＡＭ５が設けられている。ＲＯＭ４には、メインＣＰＵ１を制御するプログラムと定数データが書き込まれている。ＲＡＭ５には、静的に確保されるメモリ領域と、ヒープ領域とスタック領域のように動的に確保されるメモリ領域がある。

【００３５】続いて、メインＣＰＵ１の動作を説明する。プログラムカウンタ１７は命令アドレスを保持し、この命令アドレスにある命令を命令キャッシュ１１から命令バッファ１４に送る。このとき、命令キャッシュ１１は命令アドレスの内容を保持していない場合は、内部アドレスバス２２と内部データバス２３、バスユニット２１、外部アドレスバス２４と外部データバス２５と外部制御信号線２６、外部バス３を経由して主記憶装置２にあるＲＯＭ４から命令を読み込み、所定の形式で保持する。

【００３６】命令バッファ１４から命令を命令デコードユニット１５に転送し、命令デコード結果が制御ユニット１３を介して、レジスタファイル１６から必要ならばデータを読み出し、演算回路１８、メモリデータアクセスユニット１９、ライトバックユニット２０からなるデータパスを通り処理される。

【００３７】メモリデータアクセスユニット１９はデータキャッシュ１０に対してアドレスを生成し、当該アドレスのデータ読み出しをデータキャッシュ１０に要求し、データキャッシュ１０は所定の処理を行い、要求されたデータをメモリデータアクセスユニット１９に出力する。

【００３８】ライトバックユニット２０は、レジスタファイル１６、または、データキャッシュ１０にデータを出力する。出力先がデータキャッシュ１０の場合は、書き込むデータ、および、書き込み先アドレスをデータキャッシュ１０に出力する。

【００３９】［データの読み出し］データキャッシュ１０に当該アドレスのデータがある場合は、データキャッシュ１０が要求を処理する。データキャッシュ１０に当該アドレスのデータがない場合、当該アドレスは内部アドレスバス２２を経由してバスユニット２１に転送される。バスユニット２１は、外部アドレスバス２４と外部制御信号線２６に信号を出力して、外部バス３に要求を出し、主記憶装置２にあるＲＯＭ４またはＲＡＭ５からデータを読み出す。読み出したデータは、外部制御信号線２６の信号により外部バス３から外部データバス２５を経由してバスユニット２１に転送され、内部アドレスバス２２を経由してデータキャッシュ１０に読み込まれる。データキャッシュ１０は所定の形式でデータを保持する。

【００４０】［データの書き込み］データキャッシュ１０に書き込み先アドレスのデータが保持されている場合は、データキャッシュ１０の所定の場所にデータを書き込む。ここでは、ライトアロケート方式のキャッシュを仮定している。データキャッシュ１０に当該アドレスのデータが保持されていない場合、データキャッシュ１０にデータを保持する場所を確保し、データを書き込む。確保した場所のサイズが書き込んだデータのサイズより大きい場合は、書き込んだデータを除く部分を主記憶装置２から読み込む。

【００４１】ここでは、ライトバック方式のキャッシュを仮定している。データキャッシュ１０が主記憶装置２に書き込みを要求する場合は、当該アドレスとデータは、ライトバッファ１２に一時的に保持され、それぞれ内部アドレスバス２２と内部データバス２３を経由し、バスユニット２１に転送される。バスユニット２１は、外部アドレスバス２４と外部データバス２５と外部制御信号線２６に信号を出力して、外部バス３に要求を出し、主記憶装置２にあるＲＡＭ５に対してデータを書き込む。

【００４２】本実施の形態では、命令キャッシュ１１とデータキャッシュ１０が分離されている例を示しているが、命令キャッシュ１１とデータキャッシュ１０を融合したキャッシュにおいても、本発明を実施することができる。

【００４３】また、データキャッシュ１０と主記憶装置２の間に二次キャッシュがある場合、データキャッシュおよび二次キャッシュにおいても、本発明を実施することが可能である。

【００４４】次に、図３により、キャッシュの構成の一例を説明する。図３はキャッシュの構成を示す説明図である。

【００４５】本例は、４－ｗａｙセットアソシエイティブキャッシュである。Ｗａｙ０，Ｗａｙ１，Ｗａｙ２，Ｗａｙ３の４つのＷａｙからなり、各ＷａｙはＴａｇ４５、Ｖａｌｉｄビット（Ｖビット）４６、Ｄｉｒｔｙビット（Ｄビット）４７、データブロック４８を持つ。

【００４６】１つのＷａｙは２５６エントリからなり、各エントリは２０ビットのＴａｇ４５、１ビットのＶａｌｉｄビット４６、１ビットのＤｉｒｔｙビット４７、１２８ビットのデータブロック４８からなる。

【００４７】ＡＤＤＲＥＳＳ４０は、リードまたはライトを要求されているアドレスを示す。本例では３２ビットである。ＴＡＧ４１は、本例ではＡＤＤＲＥＳＳ４０のＭＳＢ側の２０ビットである。ＯＦＦＳＥＴ４３は、本例ではＡＤＤＲＥＳＳ４０のＬＳＢ側の４ビットである。ＩＮＤＥＸ４２は、ＴＡＧ４１とＯＦＦＳＥＴ４３の間にある８ビットである。

【００４８】ＩＮＤＥＸ４２は、インデックスデコーダ４４により、各Ｗａｙの２５６個のエントリのＩＮＤＥＸ番目の１つを指す。ＴＡＧ４１は、キャッシュエントリにあるＴａｇ４５に保持される。データブロック４８のアドレスは、キャッシュエントリにあるＴａｇ４５とＩＮＤＥＸから復元することができる。

【００４９】Ｖａｌｉｄビット４６が１にセットされている場合、当該エントリが有効であり、０にクリアされている場合は無効である。

【００５０】Ｄｉｒｔｙビット４７が１にセットされている場合、データブロック４８の内容が書き込みにより、主記憶装置の対応するメモリブロックより新しいデータに更新されていることを示す。Ｄｉｒｔｙビット４７が０にクリアされている場合は、一般には主記憶装置の対応するメモリブロックと同じデータを持つことを示す。キャッシュエントリをキャッシュから追い出すときに、Ｄｉｒｔｙビット４７が１にセットされていたらデータブロック４８のデータを主記憶装置に書き込み、０にクリアされていたらデータブロック４８のデータを主記憶装置に書き込まないで捨ててよい。

【００５１】ＬＲＵビット４９で、Ｗａｙ数をｎとすると各ＩＮＤＥＸ番目のＬＲＵビットは、

ｎ×（ｎ－１）／２

個のビット数からなる。本例は４ｗａｙなので、ＬＲＵビット４９は各ＩＮＤＥＸで、

４×３／２＝６

により６ビットからなる。つまり、ＬＲＵビット４９は４つのＷａｙで同じインデックス番号を持つエントリの４つの中から選ぶペアの組み合わせは６ペアあり、各ペアに関して新旧関係をビットで表現している。６つのビットの値からＬｅａｓｔ－Ｒｅｃｅｎｔｙ－Ｕｓｅｄ（ＬＲＵ）のエントリを見つけることができる。

【００５２】キャッシュ制御ユニット５０による、リードの場合の動作を図４に示し、ライトの場合の動作を図５に示す。このキャッシュ制御ユニット５０は、たとえば前記図２のデータキャッシュ１０の一部として構成される。図４はキャッシュのリード動作を示すフロー図、図５はキャッシュのライト動作を示すフロー図である。

【００５３】キャッシュのリード動作は、図４に示すように、まず開始後（ステップＳ１００）、ステップＳ１０１において、ＴＡＧにＡＤＤＲＥＳＳ［３１：１２］、ＩＮＤＥＸにＡＤＤＲＥＳＳ［１１：４］、ＯＦＦＳＥＴにＡＤＤＲＥＳＳ［３：０］をそれぞれ代入する。

【００５４】さらに、ステップＳ１０２において、各ｉ（ｉ＝０，１，…，Ｎ－１）で第ｉ番目のＷａｙのＩＮＤＥＸ番目のエントリに対して、Ｖａｌｉｄ（ｉ）＝１でＴａｇ＝＝ＴＡＧのとき、１→Ｈｉｔ（ｉ）、ｉ→Ｊ、Ｖａｌｉｄ（ｉ）＝０、あるいはＶａｌｉｄ（ｉ）＝１でＴａｇ＝＝ＴＡＧでないとき、０→Ｈｉｔ（ｉ）、の論理条件によりＨｉｔ（ｉ）を求める。そして、ステップＳ１０３において、求めたＨｉｔ（０），Ｈｉｔ（１），…，Ｈｉｔ（Ｎ－１）を論理和演算し、ＨＩＴに代入する。

【００５５】さらに、ステップＳ１０４において、ＨＩＴを判別し、ＨＩＴ＝１、すなわち１つでもヒットしたときは、ステップＳ１０５において、第Ｊ番目のＷａｙのＩＮＤＥＸ番目のエントリのデータブロックのＯＦＦＳＥＴにより指されているアドレスからアクセスサイズのデータを読み込む。本ブロックをｍｏｓｔ－ｒｅｃｅｎｔｌｙ－ｕｓｅｄにＬＲＵフィールドをアップデートする。

【００５６】また、ＨＩＴ＝０、すなわち１つもヒットしないときは、ステップＳ１０６において、０からＮ－１のＷａｙのＩＮＤＥＸ番目のエントリのうち、あるＫ番目のＷａｙでＶ（Ｋ）＝＝０を判別し、Ｎｏのときは、ステップＳ１０７において、全ＷａｙのＩＮＤＥＸ番目のエントリからＬＲＵブロックを選択し、これが第Ｋ番目のＷａｙにあると仮定する。

【００５７】さらに、ステップＳ１０８において、Ｄ（Ｋ）を判別し、Ｄ（Ｋ）＝１のときは、ステップＳ１０９において、第Ｋ番目のＷａｙのＩＮＤＥＸ番目のエントリのＴａｇ（Ｋ）とデータブロックおよびＩＮＤＥＸをライトバッファに待避する。当該エントリのデータブロックのＯＦＦＳＥＴに対応するワードからラップアラウンド方式で外部メモリからデータを読み込み、該当するデータがキャッシュへ到達した時点でＣＰＵへ読み

出しデータを返す。本ブロックをｍｏｓｔ－ｒｅｃｅｎｔｌｙ－ｕｓｅｄにＬＲＵフィールドをアップデートする。当該エントリで、ＴＡＧ→Ｔａｇ（Ｋ）、０→Ｄ（Ｋ）、１→Ｖ（Ｋ）のそれぞれの代入を行い、ライトバッファのデータを待避してあるＴａｇ（Ｋ）およびＩＮＤＥＸが指すアドレスに書き込む。

【００５８】また、ステップＳ１０８においてＤ（Ｋ）＝０、あるいはステップＳ１０６におけるＶ（Ｋ）＝＝０の判別の結果がＹｅｓのときは、ステップＳ１１０において、第Ｋ番目のＷａｙのＩＮＤＥＸ番目のエントリのデータブロックのＯＦＦＳＥＴに対応するワードからラップアラウンド方式で外部メモリからデータを読み込み、該当するデータがキャッシュへ到達した時点でＣＰＵへ読み出しデータを返す。本ブロックをｍｏｓｔ－ｒｅｃｅｎｔｌｙ－ｕｓｅｄにＬＲＵフィールドをアップデートする。当該エントリで、ＴＡＧ→Ｔａｇ（Ｋ）、１→Ｖ（Ｋ）、０→Ｄ（Ｋ）のそれぞれの代入を行う。

【００５９】キャッシュのライト動作は、図５に示すように、開始後（ステップＳ２００）、前記リード動作と同様に、ステップＳ２０１～Ｓ２０３において処理した後、ステップＳ２０４において、ＨＩＴを判別し、ＨＩＴ＝１のとき、ステップＳ２０６でＶ（Ｋ）＝＝０となるＷａｙ　Ｋが存在せずステップＳ２０７でＷａｙＫがＬＲＵとなりステップＳ２０８でＤ（Ｋ）＝１のとき、Ｄ（Ｋ）＝０（ステップＳ２０８）、あるいはステップＳ２０６であるＷａｙ　Ｋに対しＶ（Ｋ）＝＝０の判別の結果がＹｅｓのときは、それぞれ以下のようになる。

【００６０】ＨＩＴ＝１のときは、ステップＳ２０５において、第Ｊ番目のＷａｙのＩＮＤＥＸ番目のエントリのデータブロックのＯＦＦＳＥＴにより指されているアドレスにアクセスサイズのデータを書き込む。本ブロックをｍｏｓｔ－ｒｅｃｅｎｔｌｙ－ｕｓｅｄにＬＲＵフィールドをアップデートする。１→Ｄ（Ｊ）の代入を行う。

【００６１】ステップＳ２０８でＤ（Ｋ）＝１のときは、ステップＳ２０９において、第Ｋ番目のＷａｙのＩＮＤＥＸ番目のエントリのＴａｇ（Ｋ）とデータブロックおよびＩＮＤＥＸをライトバッファに待避する。当該エントリのデータブロックのＯＦＦＳＥＴが指すアドレスにアクセスサイズのデータを書き込み、データブロックの残りの部分へ外部メモリからデータを読み込む。本ブロックをｍｏｓｔ－ｒｅｃｅｎｔｌｙ－ｕｓｅｄにＬＲＵフィールドをアップデートする。当該エントリで、ＴＡＧ→Ｔａｇ（Ｋ）、１→Ｄ（Ｋ）、１→Ｖ（Ｋ）のそれぞれの代入を行い、ライトバッファのデータを待避してあるＴａｇ（Ｋ）およびＩＮＤＥＸが指すアドレスに書き込む。

【００６２】ステップＳ２０８でＤ（Ｋ）＝０、あるいはステップＳ２０６であるＷａｙＫに対しＶ（Ｋ）＝＝０の判別の結果がＹｅｓのときは、ステップＳ２１０において、第Ｋ番目のＷａｙのＩＮＤＥＸ番目のエントリのデータブロックのＯＦＦＳＥＴが指すアドレスにアクセスサイズのデータを書き込み、データブロックの残りの部分へ外部メモリからデータを読み込む。本ブロックをｍｏｓｔ－ｒｅｃｅｎｔｌｙ－ｕｓｅｄにＬＲＵフィールドをアップデートする。当該エントリで、ＴＡＧ→Ｔａｇ（Ｋ）、１→Ｄ（Ｋ）、１→Ｖ（Ｋ）のそれぞれの代入を行う。

【００６３】本発明は、ライトバック方式かつライトアロケート方式のキャッシュで効果がある。以下において、ライトバック方式、ライトアロケート方式を説明する。

【００６４】ライトバック方式は、ライトスルー方式と対になるキャッシュ制御方式である。ライトスルー方式では、あるアドレスにライトするときに、キャッシュに当該アドレスのエントリがあれば、そのデータブロックにライトするとともに主記憶装置の当該アドレスにもライトする。この場合、エントリのデータブロックの内容が主記憶装置の当該メモリブロックの内容と一致するので、Ｄｉｒｔｙビットを１にセットする必要はない。

【００６５】ライトバック方式では、あるアドレスにライトするときに、キャッシュに当該アドレスのエントリがあれば、そのデータブロックにライトするが、主記憶装置にはライトしない。この時、Ｄｉｒｔｙビットに１をセットしてデータブロック内容が主記憶装置の当該メモリブロックの内容と一致しないことを表す。

【００６６】ライトアロケート方式は、あるアドレスのライトにより当該アドレスが属するメモリブロックをキャッシュエントリに割り付ける方式である。リードした場合は、通常キャッシュエントリに割り付けるが、ライトの場合はライトアロケートかライトスルーかの選択の余地がある。ライトアロケート方式でライトの当該アドレスをキャッシュエントリに割り付けた場合、ライトした部分ではないデータを主記憶装置からキャッシュのデータブロックに読み込む。

【００６７】［第１の実施の形態］本実施の形態を、図６、図７および図８により説明する。図６は動的に確保される領域のキャッシュ制御方式を示すフロー図、図７は図６と図８とで使用している変数の説明図、図８はメモリ解放命令の動作を示すフロー図である。

【００６８】本実施の形態では、前述したデータキャッシュをインプリメントしたＣＰＵを考える。ライトバック方式かつライトアロケート方式とを仮定している。

【００６９】本実施の形態は、図６の動的に確保された領域のキャッシュ制御方式のアルゴリズムを実現するためのものである。本実施の形態の例では図６のステップＳ３０１の判別はコンパイラまたはプログラマにより既になされているものとする。また、図８のステップＳ４０９に示すようにスタック解放に使用できるようにレジスタの値を解放したサイズだけ加算する機能を付け加え

ている。

【００７０】すなわち、図６において、動的に確保された領域のキャッシュ制御方式は、まず開始後（ステップＳ３００）、ステップＳ３０１の動的に確保したか否かの判別処理において、動的に確保したメモリ領域を解放したか否かを判別し、解放していないとき（Ｎｏ）は終了（ステップＳ３０６）となり、解放している場合（Ｙｅｓ）は、ステップＳ３０２以降の処理に進む。

【００７１】ステップＳ３０２の初期化処理において、変数Ｓ，Ｒ，Ｅ，Ｂ，ＭＡＳＫ，ＳＳ，ＥＥに対し、解放するメモリ領域の先頭アドレス→Ｓ、解放するメモリのバイトサイズ→Ｒ、Ｓ＋Ｒ（解放するメモリ領域の終了アドレス）→Ｅ、キャッシュブロックのバイトサイズ→Ｂ、ＮＯＴ（Ｂ－１）→ＭＡＳＫ、Ｓ　ＡＮＤ　ＭＡＳＫ　＋　｛Ｂ　ｉｆ　｛Ｓ　ＡＮＤ　（Ｂ－１）｝≠０、０　ｉｆ　｛ＳＡＮＤ　（Ｂ－１）｝＝０｝→ＳＳ、Ｅ　ＡＮＤ　ＭＡＳＫ→ＥＥ、のそれぞれの代入を行う。ここでＮＯＴ　ｘは３２ビット幅でｘのビット毎の反転を表す。ｘ　ＡＮＤ　ｙは３２ビット幅でｘとｙとのビット毎の論理積を表す。

【００７２】さらに、ステップＳ３０３のループ終了判別処理において、ＳＳ＜ＥＥを判別し、Ｎｏのときは終了となり、Ｙｅｓの場合は、ステップＳ３０４のＤｉｒｔｙビットクリア処理において、ＳＳ番地に対応するキャッシュエントリのＤｉｒｔｙビットをクリアし、そしてステップＳ３０５のアドレスカウンタアップデート処理において、ＳＳ＋Ｂ→ＳＳの代入を行った後、ステップＳ３０３からの処理を繰り返す。

【００７３】ここで、図７により、前述したＳ，Ｒ，Ｅ，Ｂ，ＭＡＳＫ，ＳＳ，ＥＥの変数を具体的に説明する。

【００７４】図７においては、たとえば、解放するメモリ領域の先頭アドレスに対応する変数Ｓ＝０ｘ１００Ｃ、解放するメモリのバイトサイズに対応する変数Ｒ＝０ｘ３８、解放するメモリ領域の終了アドレスに対応する変数Ｅ＝Ｓ＋Ｒ＝０ｘ１０００Ｃ＋０ｘ３８＝０ｘ１０４４、キャッシュブロックのバイトサイズに対応する変数Ｂ＝１６、とした場合に、補正の変数ＭＡＳＫ＝ＮＯＴ（Ｂ－１）＝ＮＯＴ（０ｘＦ）＝０ｘＦＦＦＦＦＦＦ０となり、補正後の変数ＳＳ＝０ｘ１０１０、変数ＥＥ＝０ｘ１０４０となる。

【００７５】言い換えれば、Ｓ.ＳはＳ（解放するメモリ領域の先頭アドレス＝０ｘ１００Ｃ）をＢ（キャッシュブロックのバイトサイズ＝１６）の倍数に切り上げした数、ＥＥはＥ（解放するメモリ領域の終了アドレス＝０ｘ１０４４）をＢの倍数に切り捨てした数となる。すなわち、図７において、破線のメモリ領域は他で使っているかもしれないので除外する必要がある。

【００７６】本実施の形態では、スタック領域の解放をサポートする命令として、本発明を採用している。このメモリ解放命令のニモニックとオペランドを以下に示す。

【００７７】ＭＲＥＬ　Ｒｎ，ＩＭＭ

Ｒｎ：レジスタ名

ＩＭＭ：所定のビット数の即値

ＭＲＥＬ命令は、図８のメモリ解放命令ＭＲＥＬ　Ｒｎ，ＩＭＭの動作のアルゴリズムに表した動作を行う。

【００７８】今、スタック領域を関数からリターンする直前に解放する場合を考える。スタックポインタをＲ１５、解放したいメモリサイズを３２バイトとすると、

ＭＲＥＬ　Ｒ１５，３２

により、Ｒ１５←Ｒ１５＋３２を実行して、スタックの３２バイトの領域を解放するとともに、変更前のＲ１５と変更後のＲ１５で挟まれた領域のキャッシュエントリのＤｉｒｔｙビットがクリアされる。Ｄｉｒｔｙビットをクリアするときは、図８のステップＳ４０１とＳ４０２に書かれているように、解放の対象となっていないアドレスを含むキャッシュエントリは解放の対象外とするように解放の範囲を計算している。

【００７９】すなわち、図８において、メモリ解放命令ＭＲＥＬ　Ｒｎ，ＩＭＭの動作は、まず開始後（ステップＳ４００）、ステップＳ４０１の初期化処理において、変数Ｓ，Ｒ，Ｅ，Ｂ，ＭＡＳＫ，ＳＳ，ＥＥに対し、解放するメモリ領域の先頭アドレスＲｎ→Ｓ、解放するメモリのバイトサイズＩＭＭ→Ｒ、解放するメモリ領域の終了アドレス（Ｓ＋Ｒ）→Ｅ、キャッシュブロックのバイトサイズ→Ｂ、ＮＯＴ（Ｂ－１）→ＭＡＳＫ、Ｓ　ＡＮＤ　ＭＡＳＫ　＋　｛Ｂ　ｉｆ　｛Ｓ　ＡＮＤ　（Ｂ－１）｝≠０、０　ｉｆ　｛Ｓ　ＡＮＤ　（Ｂ－１）｝＝０｝→ＳＳ、Ｅ　ＡＮＤ　ＭＡＳＫ→ＥＥ、のそれぞれの代入を行う。

【００８０】さらに、ステップＳ４０２のループ終了判別処理において、ＳＳ＜ＥＥを判別し、Ｎｏのときは、ステップＳ４０９のレジスタアップデート処理において、Ｒｎ＋ＩＭＭ→Ｒｎ、の代入を行って終了（ステップＳ４１０）となり、Ｙｅｓの場合は、ステップＳ４０３以降の処理に進む。

【００８１】ステップＳ４０３のアドレス分解処理において、ＴＡＧにＳＳ［３１：１２］、ＩＮＤＥＸにＳＳ［１１：４］、ＯＦＦＳＥＴにＳＳ［３：０］をそれぞれ代入する。さらに、ステップＳ４０４のキャッシュヒット判別処理において、各ｉ（ｉ＝０，１，…，Ｎ－１）で第ｉ番目のＷａｙのＩＮＤＥＸ番目のエントリに対して、Ｖａｌｉｄ（ｉ）＝１でＴａｇ＝＝ＴＡＧのとき、１→Ｈｉｔ（ｉ）、ｉ→Ｊ、Ｖａｌｉｄ（ｉ）＝０、あるいはＶａｌｉｄ（ｉ）＝１でＴａｇ＝＝ＴＡＧでないとき、０→Ｈｉｔ（ｉ）、の論理条件によりＨｉｔ（ｉ）を求める。そして、ステップＳ４０５のキャッシュヒット集計処理において、求めたＨｉｔ（０），Ｈｉｔ（１），…，Ｈｉｔ（Ｎ－１）を論理和演算し、Ｈ

ＩＴに代入する。

【００８２】さらに、ステップＳ４０６のキャッシュヒット判別処理において、ＨＩＴを判別し、ＨＩＴ＝１、すなわち１つでもヒットしたときは、ステップＳ４０７のＤｉｒｔｙビットクリア処理において、第Ｊ番目のＷａｙのＩＮＤＥＸ番目のエントリに対し０→Ｄｉｒｔｙ（Ｊ）の代入を行った後、ステップＳ４０８に進む。

【００８３】また、ＨＩＴ＝０、すなわち１つもヒットしないときは、ステップＳ４０８のアドレスカウンタアップデート処理において、ＳＳ＋Ｂ→ＳＳの代入を行った後に、ステップＳ４０２からの処理を繰り返して実行する。

【００８４】なお、マルチタスク方式でスタック領域がタスク毎に排他的に割り付けられている場合も、本実施の形態の例をそのまま適用できる。

【００８５】［第２の実施の形態］本実施の形態は、前記第１の実施の形態の変形例である。前記第１の実施の形態で示した、ＭＲＥＬ命令に指定するレジスタはスタックポインタに使われているレジスタだけとは限らない。ヒープ領域に動的に確保したメモリ領域を解放するときに使うことができる。

【００８６】具体的には、Ｃ言語の標準ライブラリ関数にあるｆｒｅｅ関数の内部で解放した領域に対してＭＲＥＬ命令を施す例がある。解放する領域の先頭アドレスをＲ１、解放するバイト数を１００とすると、

ＭＲＥＬ　Ｒ１，１００

を実行することにより、解放した領域に対応するキャッシュエントリのダーティビットがクリアされる。

【００８７】［第３の実施の形態］本実施の形態を、図９により説明する。図９はＤｉｒｔｙビットクリア命令の動作を示すフロー図である。

【００８８】本実施の形態では、前記図６のステップＳ３０４をＣＰＵの命令として実現したものである。Ｄｉｒｔｙビットクリア命令のニモニックとオペランドを以下に示す。

【００８９】ＤＣＢＤＣ　@Ｒｎ

Ｒｎ：レジスタ名

ＤＣＢＤＣ命令は、図９のＤｉｒｔｙビットクリア命令ＤＣＢＤＣ　@Ｒｎの動作のアルゴリズムに表した動作を行う。

【００９０】すなわち、図９において、Ｄｉｒｔｙビットクリア命令ＤＣＢＤＣ　@Ｒｎの動作は、まず開始後（ステップＳ５００）、ステップＳ５０１の対象アドレス取得処理において、ＳＳにＲｎ（アドレス）を代入する。

【００９１】さらに、ステップＳ５０２のアドレス分解処理において、ＴＡＧにＳＳ［３１：１２］、ＩＮＤＥＸにＳＳ［１１：４］、ＯＦＦＳＥＴにＳＳ［３：０］をそれぞれ代入する。さらに、ステップＳ５０３のキャッシュヒット判別処理において、各ｉ（ｉ＝０，１，…，Ｎ－１）で第ｉ番目のＷａｙのＩＮＤＥＸ番目のエントリに対して、Ｖａｌｉｄ（ｉ）＝１でＴａｇ＝＝ＴＡＧのとき、１→Ｈｉｔ（ｉ）、ｉ→Ｊ、Ｖａｌｉｄ（ｉ）＝０、あるいはＶａｌｉｄ（ｉ）＝１でＴａｇ＝＝ＴＡＧでないとき、０→Ｈｉｔ（ｉ）、の論理条件によりＨｉｔ（ｉ）を求める。そして、ステップＳ５０４のキャッシュヒット集計処理において、求めたＨｉｔ（０），Ｈｉｔ（１），…，Ｈｉｔ（Ｎ－１）を論理和演算し、ＨＩＴに代入する。

【００９２】さらに、ステップＳ５０５のキャッシュヒット判別処理において、ＨＩＴを判別し、ＨＩＴ＝１、すなわち１つでもヒットしたときは、ステップＳ５０６のＤｉｒｔｙビットクリア処理において、第Ｊ番目のＷａｙのＩＮＤＥＸ番目のエントリに対し０→Ｄｉｒｔｙ（Ｊ）の代入を行って終了（ステップＳ５０７）となる。また、ＨＩＴ＝０、すなわち１つもヒットしないときは、終了となる。

【００９３】従って、前記実施の形態によれば、主記憶装置の解放した領域に対応するキャッシュエントリのＤｉｒｔｙビットをクリアすることにより、このＤｉｒｔｙビットをクリアしたキャッシュエントリがキャッシュに残っている場合に、同一のアドレス範囲にある主記憶装置の領域を新たに確保し、この領域に最初にデータを書き込んだ時に、既に当該アドレス範囲に対応するキャッシュエントリがキャッシュに存在するため、ライトアロケートする必要がなく、データ転送が発生しない。よって、データ転送のための電力消費を発生せず、データ転送の経路を混雑させて性能を劣化させることもない。

【００９４】また、ＤｉｒｔｙビットをクリアしたキャッシュエントリがＬＲＵアルゴリズムなどによりキャッシュから追い出される場合においても、Ｄｉｒｔｙビットがクリアされているために、キャッシュエントリのデータを主記憶装置に書き戻す必要がなく、書き戻しのためのデータ転送が発生しない。よって、データ転送のための電力消費を発生せず、データ転送の経路を混雑させて性能を劣化させることもない。

【００９５】以上、本発明者によってなされた発明をその実施の形態に基づき具体的に説明したが、本発明は前記実施の形態に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能であることはいうまでもない。

【００９６】

【発明の効果】本願において開示される発明のうち、代表的なものによって得られる効果を簡単に説明すれば、以下のとおりである。

【００９７】（１）ダイナミックに確保した主記憶装置の領域を解放するときに、対応するキャッシュエントリのＤｉｒｔｙビットをクリアすることで、再度同一領域を確保した場合に、同一キャッシュエントリにライトアロケートが発生しないので、ライトアロケートによるキ

ャッシュメモリと主記憶装置間のデータ転送をなくすことが可能となる。

【００９８】（２）前記（１）により、主記憶装置とキャッシュメモリとの間の不要なデータ転送を削減することができるので、このデータ転送に伴う電力消費を削減し、データ転送の経路の混雑を緩和してスループットを向上させることが可能となる。

【図面の簡単な説明】

【図１】（ａ），（ｂ）は本発明の一実施の形態の携帯電話システムを示す説明図である。

【図２】本発明の一実施の形態において、メインＣＰＵとＲＯＭとＲＡＭからなる部分のキャッシュメモリシステムを示すブロック図である。

【図３】本発明の一実施の形態において、キャッシュの構成を示す説明図である。

【図４】本発明の一実施の形態において、キャッシュのリード動作を示すフロー図である。

【図５】本発明の一実施の形態において、キャッシュのライト動作を示すフロー図である。

【図６】本発明の一実施の形態において、動的に確保される領域のキャッシュ制御方式を示すフロー図である。

【図７】本発明の一実施の形態において、変数の説明図である。

【図８】本発明の一実施の形態において、メモリ解放命令の動作を示すフロー図である。

【図９】本発明の一実施の形態において、Ｄｉｒｔｙビットクリア命令の動作を示すフロー図である。

【図１０】本発明の前提として検討した技術において、スタックの使用方法を示す説明図である。

【図１１】本発明の前提として検討した技術において、キャッシュの状態を示す説明図である。

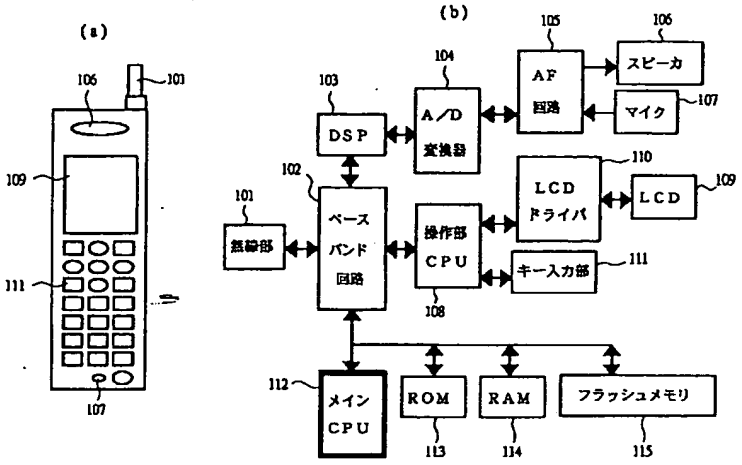
【図１２】（ａ）～（ｄ）は本発明の前提として検討した技術および本発明の技術において、ＩＮＤＥＸが０ｘＡ８のキャッシュエントリを示す説明図である。

【符号の説明】

１　メインＣＰＵ
２　主記憶装置
３　外部バス
４　ＲＯＭ
５　ＲＡＭ
６　スタックポインタ
１０　データキャッシュ
１１　命令キャッシュ
１２　ライトバッファ
１３　制御ユニット
１４　命令バッファ
１５　命令デコードユニット
１６　レジスタファイル
１７　プログラムカウンタ
１８　演算回路
１９　メモリデータアクセスユニット
２０　ライトバックユニット
２１　バスユニット
２２　内部アドレスバス
２３　内部データバス
２４　外部アドレスバス
２５　外部データバス
２６　外部制御信号線
４０　ＡＤＤＲＥＳＳ
４１　ＴＡＧ
４２　ＩＮＤＥＸ
４３　ＯＦＦＳＥＴ
４４　インデックスデコーダ
４５　Ｔａｇ
４６　Ｖａｌｉｄビット
４７　Ｄｉｒｔｙビット
４８　データブロック
４９　ＬＲＵビット
５０　キャッシュ制御ユニット
６０　ＡＤＤＲＥＳＳ
６１　ＴＡＧ
６２　ＩＮＤＥＸ
６３　ＯＦＦＳＥＴ
６４　Ｗａｙ　Ｊ
１０１　無線部
１０２　ベースバンド回路
１０３　ＤＳＰ
１０４　Ａ／Ｄ変換器
１０５　ＡＦ回路
１０６　スピーカ
１０７　マイク
１０８　操作部・ＣＰＵ
１０９　ＬＣＤ
１１０　ＬＣＤドライバ
１１１　キー入力部
１１２　メインＣＰＵ
１１３　ＲＯＭ
１１４　ＲＡＭ
１１５　フラッシュメモリ

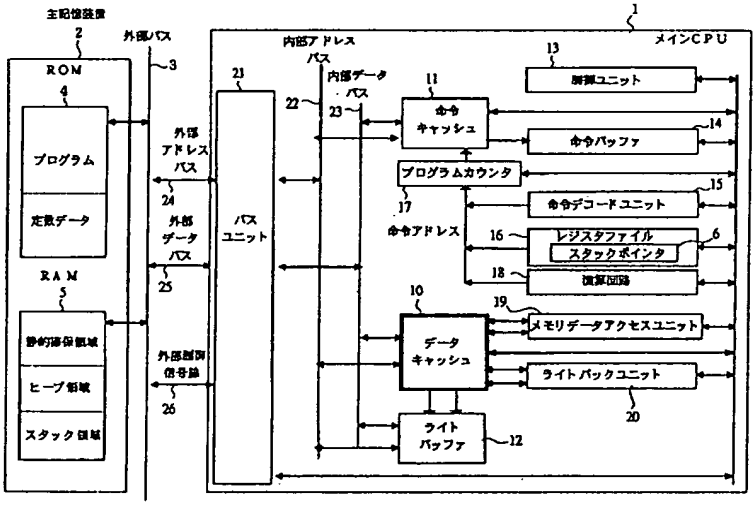
【図１】

図 1



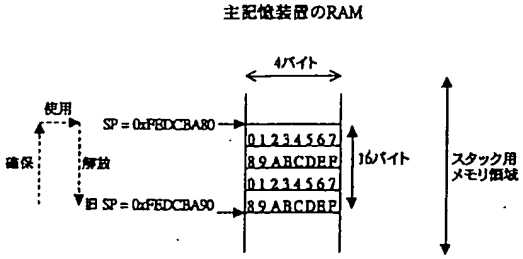
【図２】

図 2



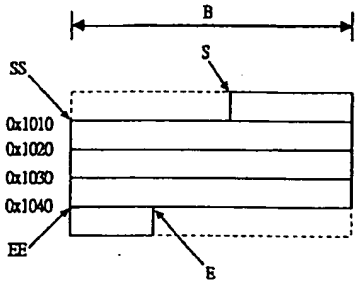
【図１０】

図 10



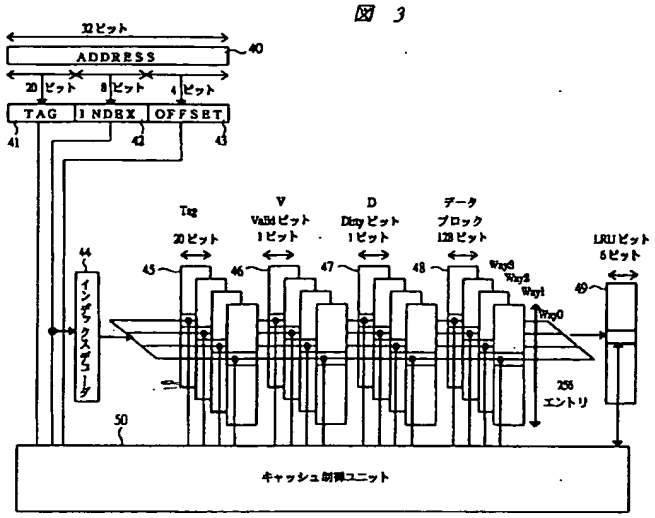
【図７】

図 7



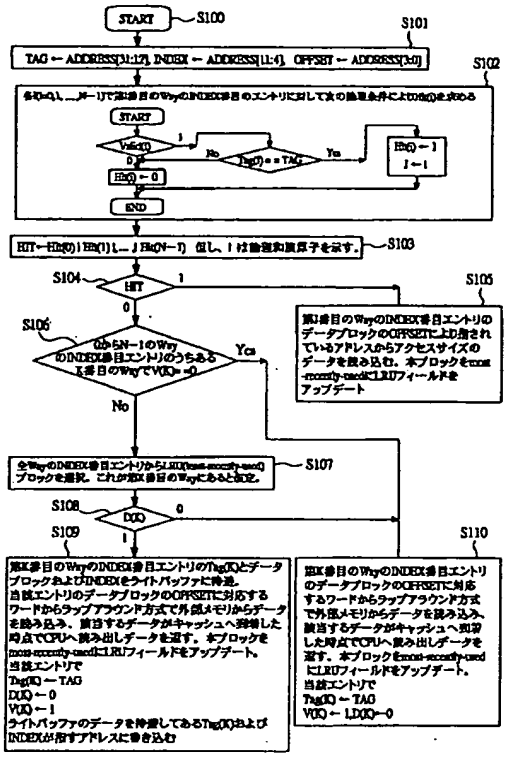
【図３】

図 3



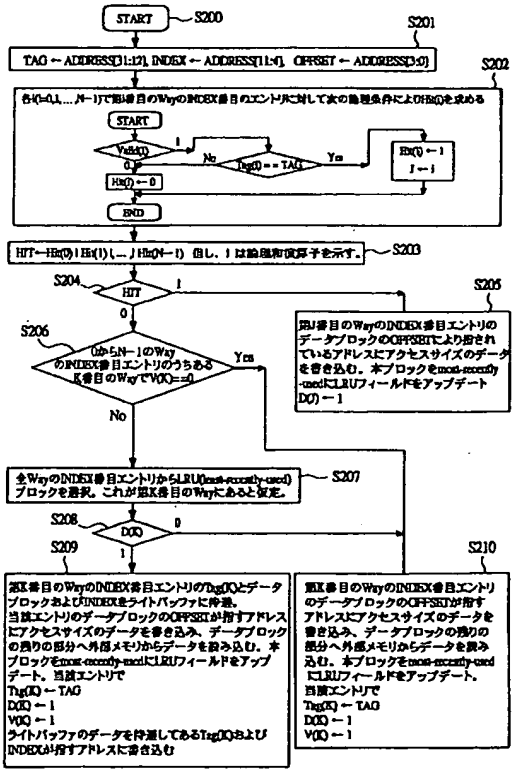
【図４】

図 4



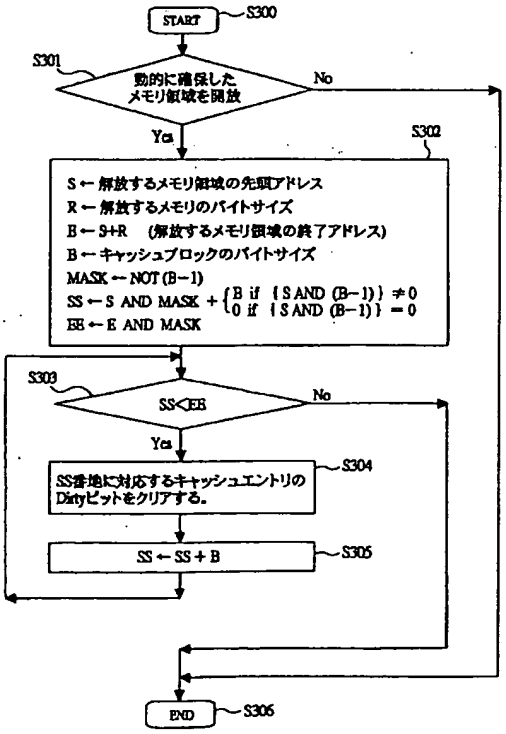
【図５】

図 5



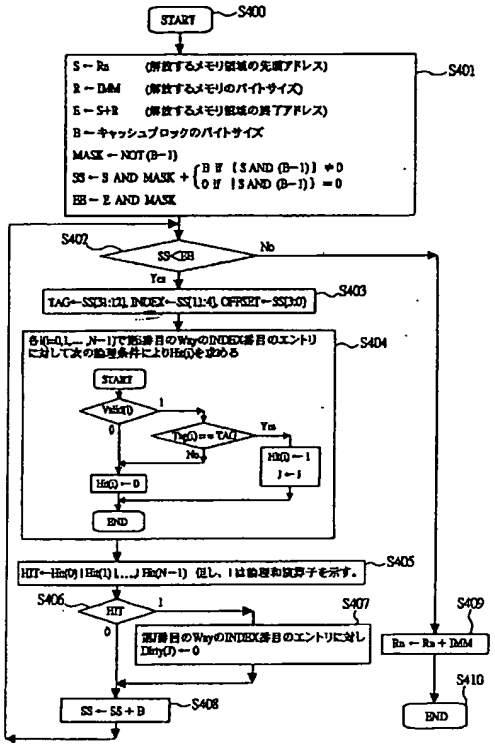
【図６】

図 6



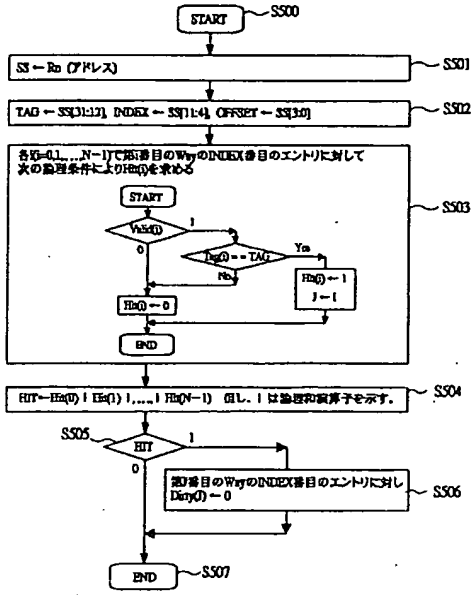
【図８】

図 8



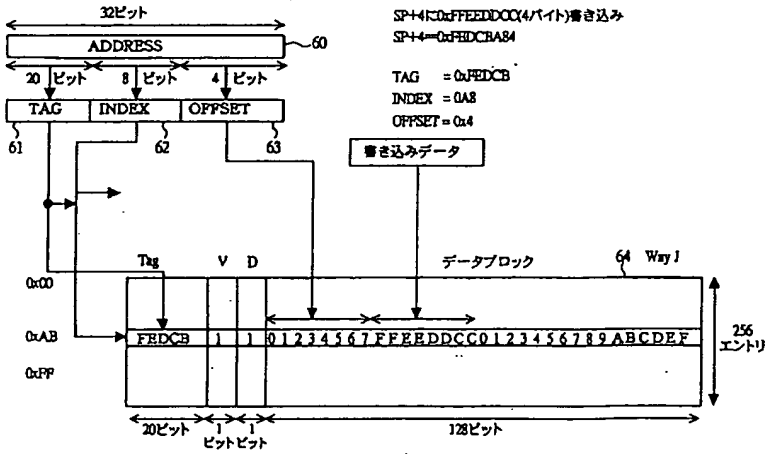
【図９】

図 9



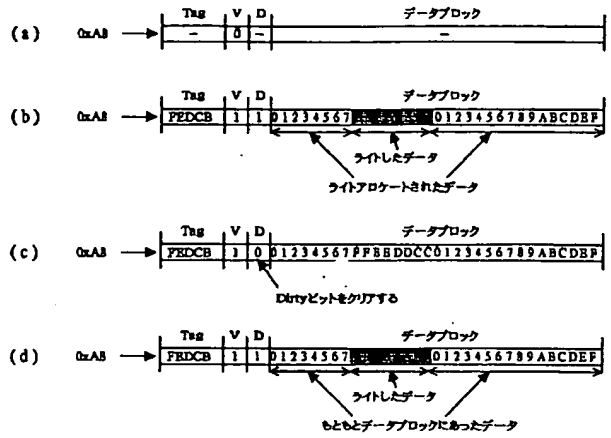
【図１１】

図 11



【図１２】

図 12



フロントページの続き

(51)Int.Cl.'		識別記号	FI		テーマコード'(参考)
Ｇ０６Ｆ	12/08	５７９	Ｇ０６Ｆ 12/08	５７９	
	12/12	５０３	12/12	５０３	

Japanese Laid-Open Patent Application No. 2003-223360

[What is claimed is]

[Claim 1] A cache memory system comprising
a write back method cache memory,
wherein said cache memory system can execute a command for manipulating a dirty bit of a reserved area in said cache memory.

[Claim 2] The cache memory system according to Claim 1,
wherein the command is a memory releasing command, and
a mnemonic and operand of the memory release command are:
MREL Rn, IMM
Rn: Register name
IMM: immediate value having a predetermined number of bits.

[Claim 3] The cache memory system according to Claim 1,
wherein the command is a dirty bit clearing command, and
the mnemonic and operand of the dirty bit clearing command are:
DCBDC @Rn
Rn: Register name.

[Claim 4] The cache memory system according to one of Claim 2 and Claim 3,
wherein, the Rn: Register name is a register that can be used for releasing a dynamically reserved memory area in a stack area.

[Claim 5] The cache memory system according to one of Claim 2 and Claim 3,
wherein, the Rn: Register name is a register that can be used for releasing a dynamically reserved memory area in a heap area.

[Claim 6] The cache memory system according to Claim 1, further comprising
a processor operable to instruct execution of the command; and
a main storage device,
wherein a region identified by a specific address in said main storage device is reserved and a predetermined program process is performed, based on an instruction from said processor, and

when, after the reserved area is reserved in said cache memory and used, the reserved area in said main storage device is to be cleared according to an instruction from said processor, control is performed so as to clear a dirty bit of the region reserved in said cache memory.

[Claim 7] A microprocessor comprising:

a write back method cache memory,

wherein said cache memory system can execute a command for manipulating a dirty bit of a reserved area in said cache memory.

[0001]

[Field of the Invention]

The present invention relates to cache memory control technology, and particularly to effective technology applied to a cache memory control method for releasing a dynamically reserved memory area in a cache memory system that has a cache memory, and the like, of a micro processor, a micro computer, or a micro controller.

[0002]

[Description of the Related Art]

According to the consideration of the inventor, cache memory control technology is cited in technology described in the respective publications of Japanese Laid-Open Patent Application No.11-338772, Japanese Laid-Open Patent Application No.4-268638, Japanese Laid-Open Patent Application No.4-264641, Japanese Laid-Open Patent Application No.4-188326, Japanese Laid-Open Patent Application No.6-28253, and Japanese Laid-Open Patent Application No.9-231133.

[0003]

In the above-mentioned technology included in the publications, when in the condition in which an area reserved dynamically in the main memory is read as a cache memory entry, such area of the main memory is released, such cache entry is made invalid by clearing a valid bit of the cache entry.

[0004]

[Problems that the Invention is to Solve]

As a result that the inventor has considered a cache memory control technology

such as the above-mentioned technology, the following has become apparent. The following describes the technology that the inventor has examined as the prerequisites of the present invention with reference to Figure 10 to 12. Figure 10 is an explanatory diagram of the stack usage, Figure 11 the cache condition, and Figure 12 the cache entry with an INDEX as 0xA8.

[0005]

[Description of General Stack Usage: Figure 10]

Assume that the stack pointer SP before using has the value of SP=0xFEDCBA90. 0x is a prefix which indicates that the successive character string is hexadecimal. When reserving a 0x10-byte area in the stack, SP←SP-0x10 is performed, whereby SP=0xFEDCBA80 is obtained.

[0006]

The 0x10-byte memory area between User Old SP=0xFEDCBA90 and Current SP=0xFEDCBA80 is available. For example,

Data 0x01234567 is written to the address (SP+0);

Data 0x89ABCDEF is written to the address (SP+4);

Data 0x01234567 is written to the address (SP+8);

Data 0x89ABCDEF is written to the address (SP+12).

When these data become unnecessary and a memory area for these data no longer needs to be reserved, SP←SP+0x10 is performed. Then, SP returns to the value of Old SP and the reserved 0x10-byte memory area is released.

[0007]

Memory areas in addresses lower than the SP in stack memory areas are unused and released. Among stack memory areas, memory areas that have higher addresses than the SP are already reserved.

[0008]

[Cache operation during writing: Figure 11]

Figure 11 shows the cache condition after data 0xFFEEDDCC is written into the address (SP+4), that is, into the address 0xFEDCBA84.

[0009]

The address 0xFEDCBA84 to be written is treated as ADDRESS60, and the 20 bits

from the MSB of the ADDRESS is treated as TAG 61, the next 8 bits as INDEX 62 and the last 4 bits as OFFSET 63. It is assumed that the entry of INDEX=0 x A8 of Way J 64 is in the Invalid condition (Valid bit V=0) before being written and the Way J was selected for this writing operation. In this entry, when V=0, although undefined values are included in Tag, Dirty bit (D bit) and a data block, 0 x FEDCB is written into Tag, 1 in the V bit, 1 in the D bit, and data 0xFFEEDDCC in 4 bytes from the position that the data block offset is OFFSET=4.

[0010]

Since this description assumes a cache memory based on the write-back method, when there is no coherence between the main memory and the cache entry's data block, D bit is set to 1. Data from the corresponding addresses of the main memory can be read in the rest of this data block by using the write allocate feature. Figure 11 shows this condition.

[0011]

[Specific Description of Prerequisite Technology: Figure 12]

In the prerequisite technology of the present invention, SP←SP+0x10 releases the stack memory area of 0x10 bytes and at the same time the valid bit (V bit) of the cache entry allocated with a released memory area is cleared to 0 to set Invalid. In the case that V bit is not cleared, when this cache entry is evicted from the cache memory according to the LRU algorithm, if D bit is 1, the content of the data block is written back to the main memory. However, because the content of this data block is basically the data of an area that has been already released, it is not reused even if it is written back to the main memory. So such write back is useless. Therefore, useless write back has been prevented by clearing V bit in order to avoid write back. Figure 12 (a) shows the Invalid condition of the cache entry.

[0012]

[Problems of the Prerequisite Technology of the Invention]

The problem of the above-mentioned prerequisite technology of the present invention is revealed when the address, which is the same as that of an area that was released once, is reserved again in the write-allocate method cache memory. In the prerequisite technology of the present invention, because the Valid bit is cleared when the area is released, the cache entry becomes invalid and the tag and data are lost. When the same address range is reserved again and is written again, new allocation of a cache entry is

necessary because the tag has already been lost. According to the write-allocate method, the cache entry data, except the written data, is read into the cache entry from the main memory. This data transfer may cause power consumption and the congestion of data routes, leading to performance deterioration. Solving these problems is the object of the present invention.

[0013]
Assuming that when a 16-byte stack area beginning with 0xFEDCBA80 is released, the cache entry is set to Invalid as shown in Figure 12 (a) based on the prerequisite technology of the present invention. Next, when it is assumed that the value of 0xBBBBAAAA is written into the address of: 0xFEDCBA84, which is allocated to the same Way, the result is shown in Figure 12 (b). In this case, the part other than the 4-byte written data of the data block is read from the main memory. The object of the present invention is to reduce power consumption and eliminate the congestion of data transfer routes caused by this reading operation, and thus improving throughput.

[0014]
In other words, the object of the present invention is to reduce unnecessary data transfer between the main memory and the cache memory as well as power consumption along with data transfer, and alleviate the congestion of data transfer routes, improving throughput.

[0015]
The above object of the present invention as well as other objects and new features shall become clear with reference to the description specification and attached figures.

[0016]
[Means to Solve the Problems]
The overview of some typical inventions to be disclosed in the present invention is briefly described hereinafter.

[0017]
The cache memory system according to the present invention is a cache memory system including a write back cache memory, wherein said cache memory system can execute a command for manipulating a dirty bit in said cache memory. In particular, the command is a memory releasing command MREL Rn, IMM, or a dirty bit clearing

command DCBDC @Rn, which is applied in a register which can be used for releasing a dynamically reserved memory area in a stack area or a heap area.

[0018]
In addition, in the above-mentioned cache memory system, a processor directing the implementation of an order and the main memory are included. Based on the direction of the processor, an area indicated by the specific address within the main memory is reserved and the predetermined program is executed. After this reserved area is reserved on the cache memory and is used, when the reserved area in the main memory is released according to the direction of the processor, the Dirty bit of the area reserved on the cache memory is controlled so as to be cleared.

[0019]
In other words, while, in the prerequisite technology of the present invention, the cache entry Valid bit corresponding to the released area of the main memory is cleared, the present invention is characterized in that the cache entry Dirty bit is cleared.

[0020]
Consequently, in the case where the cache entry in which the Dirty bit is cleared is present in the cache, a main memory area in the same address range is newly reserved and, when data is first written to this area, the effects of the present invention are seen. As the cache entry corresponding to the address range is already present in the cache, there is no need for allocating data to be written and no data transfer takes place. Therefore, there is no power consumption caused by data transfer and no performance deterioration caused by the congestion of data transfer routes.

[0021]
In addition, in the case that the cache entry in which the Dirty bit is cleared is evicted from the cache, because the Dirty bit is cleared, the cache entry data need not be written back to the main memory and no data transfer takes place for the write back. Therefore, there is no power consumption caused by data transfer and no performance deterioration caused by the congestion of data transfer routes.

[0022]
For example, in the condition where a 16-byte stack area beginning with 0xFEDCBA80 is allocated to the cache entry of INDEX=0xA8 of Way J as shown in the

above-mentioned Figure 11, when the area is released, in the present invention, the cache entry Dirty bit is set to 0 as shown in Figure 12 (c). If the value of 0xBBBAAAA is written to the address of 0xFEDCBA84 before this cache entry is evicted from the cache, the condition of Figure 12 (d) is provided. The part other than the written data of the data block is never read from the main memory and the part is the value originally written in the data block. Therefore, although in the prerequisite technology of the present invention, reading from the main memory takes place based on the write-allocate method, this reading does not take place in this method. Power consumption caused by this reading is reduced and the congestion of data transfer routes caused by this read-in does not take place, resulting in the improvement of throughput.

[0023]
In addition, when the cache entry in the condition of Figure 12 (c) is evicted from the cache based on the LRU algorithm, because the Dirty bit is cleared to 0, there is no power consumption caused by data transfer for write back and no performance deterioration caused by the congestion of data transfer routes.

[0067]
[First Embodiment]
The embodiment is described with reference to Figure 6, 7 and 8. Figure 6 is a flow chart indicating the cache control method of the dynamically reserved area; Figure 7 describes variables used in Figure 6 and 7; Figure 8 is a flow chart indicating the order behavior for releasing a memory.

[0068]
In the present embodiment, a CPU in which the above-mentioned data cache is implemented is discussed. The write-back method and the write-allocate method are assumed to be used.

[0069]
The description of the embodiment is for realizing the algorithm of the cache control method in the dynamically reserved area as shown in Figure 6. In the example of the description of the embodiment, the judgment in step S 301 of Figure 6 is assumed to have already been performed by a compiler or a programmer. In addition, as shown in step S 301 of Figure 6, a function is added to add the register value enough to make up for the released value in order to use it for the release of the stack.

[0070]
In other words, in Figure 6, soon after the start (step S 300), the cache control method of a dynamically reserved area judges if a dynamically reserved memory area is released or not in the judgment process to judge the dynamically reserved area in step S 301. When the memory area is not released (No), the process is terminated (step S 306) and when the memory area is released (Yes), the process moves on to step S 302 and the successive steps.

[0071]
In the initialization process of step S 302, the following assignment is done for variables including S, R, E, B, MASK, SS, and EE: the initial address of a memory area to be released → S; the size of a memory area to be released → R; S + R (the ending address of a memory area to be released) → E; the byte size of a cache block → B; NOT (B-1) → MASK; S AND MASK+{B if {S AND (B-1)}≠0, 0 if {SAND (B-1)}=0} → SS; E AND MASK → EE. NOT x represents an inversion per bit of x at a 32 bit width. x AND y represents the conjunction per bit of x and y at a 32 bit width.

[0072]
In addition, in the discrimination process of the loop termination of step S 303, if the discrimination of SS<EE is NO, the process is terminated. If the discrimination is YES, a cache entry Dirty bit corresponding to the address SS is cleared in the Dirty bit clear process. Subsequently, after SS + B is assigned to SS in the address counter update process of step S 305, step S 303 and the successive processes are repeated.

[0073]
Figure 7 describes the above-mentioned variables such as S, R, E, B, MASK, SS, and EE.

[0074]
In Figure 7, for example, if the variable corresponding to the initial address of a memory area to be released is S=0x100 C, the variable corresponding to the byte size of a memory to be released is R = 0 x 38, the variable corresponding to the ending address of a memory area to be released is E=S+R=0x1000C+0x38=0x1044, and the variable corresponding to the byte size of a cache block is B=16, the correction variable MASK= NOT(B-1)=NOT(0xF)=0xFFFFFFF0, and the variable after correction SS=0x1010 and

the variable EE=0x1040 are provided.

[0075]
In other words, SS is the number that S (the initial address of a memory area to be released =0x100C) is rounded out to the multiple number of B (the cache block byte size =16), and EE is the number that E (the ending address of a memory area to be released =0x1040) is truncated to the multiple number of B. Namely, in Figure 7, because a dashed lined memory area may be used for other processes, it needs to be excluded.

[0076]
In the description of the embodiment, the present invention is applied as the instruction for supporting the release of a stack area. The mnemonic and operand of this memory release order are shown as follows.

[0077]
MREL Rn, IMM
Rn: register name
IMM: the predetermined immediate bit number
MREL order implements the behavior represented in the behavior algorithm of the memory release order MREL, Rn, and IMM of Figure 8.

[0078]
For example, in the case of releasing a stack area immediately before returning it from a function, when the stack pointer is R15 and a memory size to be released is 32 bytes, according to MREL R15, 32, R15←R15+32 is performed, releasing a 32 byte area of the stack. At the same time, the Dirty bit of a cache entry area between the pre-change R15 and the post-change R15 is cleared. When Dirty bit is cleared, as described in step S 401 and 402 of Figure 8, the release range is calculated, disregarding a cache entry including an address that is not the release target.

[0079]
In other words, in Figure 8, the behavior of a memory release order MREL Rn, IMM, is the implementation of the following assignment to variables including S, R, E, B, MASK, SS, and EE in the initialization process of step S 401 after the start (step S 400): the initial address of a memory area to be released, Rn → S; the byte size of a memory

area to be released, IMM → R; the ending address of a memory area to be released, (S+R) → E; the byte size of a cache block → B; NOT (B-1) → MASK; S AND MASK +{B if {S AND (B-1)}≠0, 0 if {SAND (B-1)}=0} → SS; E AND MASK → EE.

[0080]
In addition, in the discrimination process of the loop termination of step S 402, if the discrimination of SS<EE is No, the assignment of Rn + IMM → Rn is performed in the register update process of step S 409 and the process is terminated (step S 410). If the discrimination is Yes, the process moves on to step S 403 and the successive processes.

[0081]
In the address decomposition process of step S 403, SS [31:12], SS [11:4] and SS[3:0] are inputted to TAG, INDEX and OFFSET respectively. In addition, in the cache hit discrimination process of step S 404, Hit (i) is calculated according to the logic requirement that when Valid (i)=1 and Tag = = TAG, 1 → Hit (i), i → J, Valid (i)=0, or Valid (i)=1 but not Tag = = TAG, 0 → Hit (i) is provided for the INDEXth entry of the ith Way at each i (i = 0, 1, ..., N - 1). Then, in the cache hit calculation process of step S 405, the logical OR of the calculated Hit (0), Hit (1), ..., Hit (N-1) is calculated and assigned to IT.

[0082]
In addition, in the cache hit discrimination process of step S 406, after HIT is discriminated, when HIT = 1, namely there is a hit, 0 → Dirty (J) is assigned to the INDEXth entry of the Jth Way in the Dirty bit clear process of step S 407. Then the process moves on to step S 408.

[0083]
In addition, when HIT=0, namely there is no hit, after the assignment of SS+B → SS is performed in the address counter update process, step S 402 and the successive processes are repeatedly performed.

[0084]
In addition, in the case of assigning a stack area exclusively allocated to each task based on the multitask method, the examples of the description of the embodiment can be applied without modification.

[0085]

[Second Embodiment]

The description of the embodiment is a deformation example of the above-mentioned First Embodiment. As shown in the above-mentioned First Embodiment, a register to be specified to the MREL order is not limited to a register used for a stack pointer. It can be used when a dynamically reserved memory area in a heap area is released.

[0086]

Specifically, there is an example that the MREL order is implemented for an area released inside the free function in the standard library functions of the C language. If the initial address of an area to be released is R1 and the byte number to be released is 100, the Dirty bit hit of a cache entry corresponding to the released area is cleared by implementing MREL R1, 100.

[0087]

[Third Embodiment]

The description of the embodiment is shown with reference to Figure 9. Figure 9 is a flow chart indicating the behavior of the Dirty bit clear order.

[0088]

In the description of the embodiment, step S 304 of the above-mentioned Figure 6 is realized as the order of a CPU. The mnemonic and operand of the Dirty bit clear order are shown as follows.

[0089]

DCBDC @Rn

Rn: register name

The DCBDC order implements the behavior represented in the behavior algorithm of DCBDC @Rn, the Dirty bit clear order of Figure 9

[0090]

In other words, in Figure 9, after the start (step S 500), the behavior of the Dirty bit clear order DCBDC @Rn assigns Rn (address) to SS in the process of obtaining the target address of step S 501.

[0091]

In addition, in the address decomposition process of step S 502, SS [31:12], SS [11:4] and SS[3:0] are assigned to TAG, INDEX and OFFSET respectively. Moreover, in the

cache hit discrimination process of step S 503, Hit (i) is calculated according to the logic requirement that when Valid (i)=1 and Tag = = TAG, 1 → Hit (i), i → J, Valid (i)=0, or Valid (i)=1 but not Tag = = TAG, 0 → Hit (i) is provided for the INDEXth entry of the ith Way at each i (i = 0, 1, ..., N-1). Then, in the cache hit calculation process of step S 504, the logical OR of the calculated Hit (0), Hit (1), ..., Hit (N-1) is calculated and assigned to IT.

[0092]
In addition, in the cache hit discrimination process of step S 505, after HIT is discriminated, when HIT = 1, namely there is a hit, 0 → Dirty (J) is assigned to the INDEXth entry of the Jth Way in the Dirty bit clear process of step S 506 and the process is terminated (step S 507). When HIT = 0, namely there is no hit, the process is terminated.

[0093]
Therefore, according to the above-mentioned description of the embodiment, because a cache entry Dirty bit corresponding to the released area of the main memory is cleared, in the case that the cache entry in which this Dirty bit was cleared is present in the cache, the main memory area in the same address range is newly reserved. When data is first written to this area, because a cache entry corresponding to the address range is present in the cache, there is no need of allocating data to be written and no data transfer takes place. Thus, there is no power consumption caused by data transfer, and no performance deterioration caused by the congestion of the data transfer routes.

[0094]
In addition, in the case that the cache entry which cleared the Dirty bit is evicted from the cache based on The LRU algorithm, because the Dirty bit is cleared, the cache entry data is not needed to be written back to the main memory and there is no data transfer for writing back. Therefore, power for data transfer is not consumed and there is no performance deterioration caused by the congestion of data transfer routes.

[0095]
So far the present invention by the inventor has been described based on the description of the embodiment. However, the present invention is not limited to the above-mentioned description of the embodiment and it goes without saying that various changes are possible without departing from the scope of the present invention.

[0096]

[Effects of the present invention]

The effects of some typical inventions to be disclosed in the claim are described as follows:

[0097]

(1) In the case that when the dynamically reserved main memory area is released, the same area is reserved again by clearing the corresponding cache entry Dirty bit, write-allocate does not take place in the same cache entry, allowing the removal of data transfer between the cache memory and the main memory for write-allocate.

[0098]

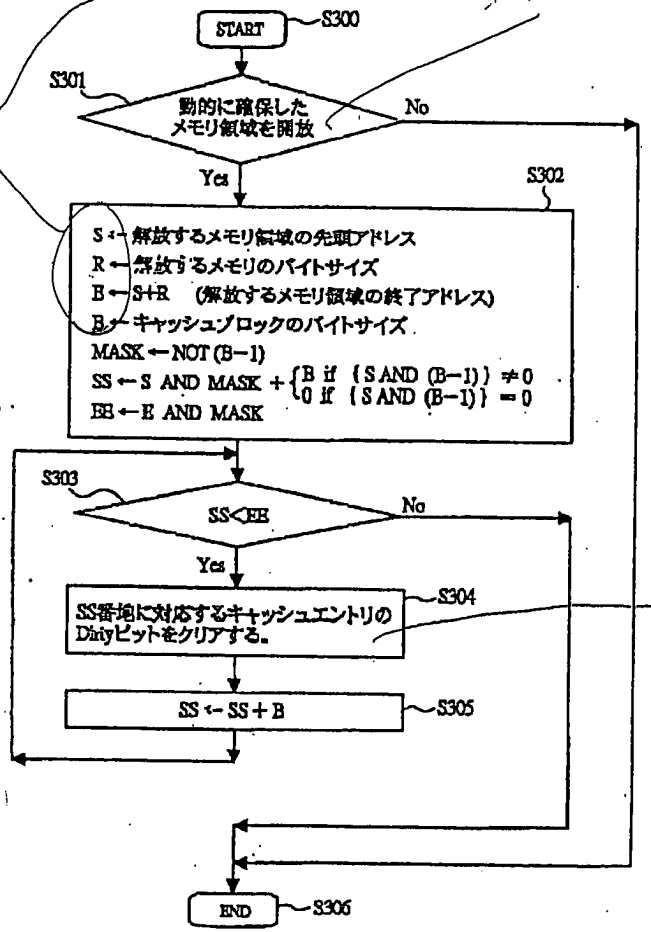
(2) According to the above-mentioned (1), since unnecessary data transfer between the main memory and the cache memory can be reduced, power consumption along with data transfer is reduced and the congestion of data transfer routes is alleviated, improving throughput.

Figure 6

【図6】

図6

Releasing the dynamically reserved memory area



The cache entry Dirty bit corresponding to the address ss is cleared.

+ Initial address of a memory area to be released

= Byte size of a memory to be released

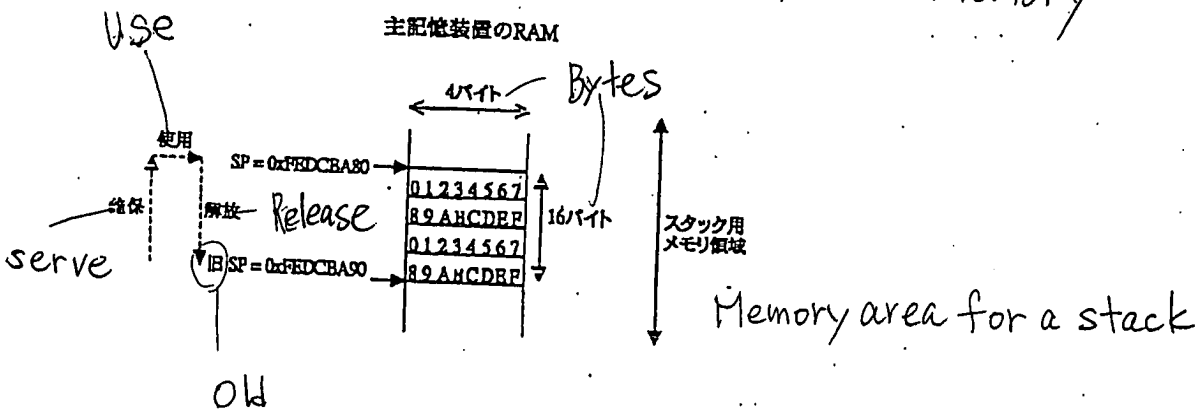
= S+R (= Ending address of a memory area to be released)

= Cache block byte size

Figure 10

【図１０】

図 10



RAM of Main Memory

Memory area for a stack

Figure 7

【図７】

図 7

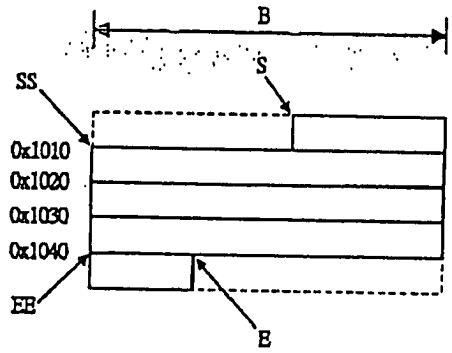


Figure 8

【図8】

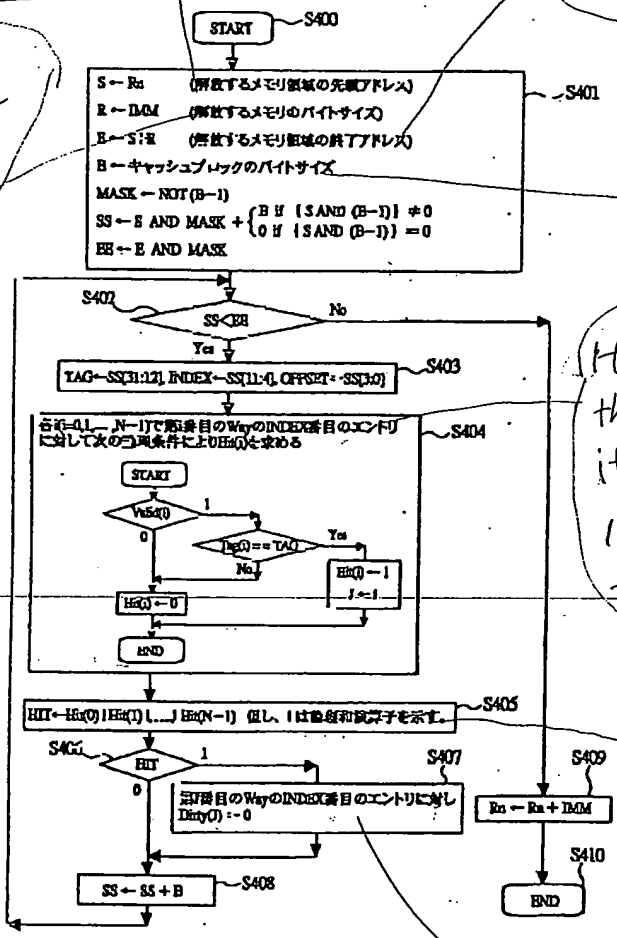
図 8

- Initial address of a memory area to be released

Ending address of a memory area to be released

Byte size of a memory to be released

Byte size of a cache block



Hit(i) is calculated for the INDEXth entry of the ith way at each i (i = 0, 1, ..., N-1) according to the following logic requirement

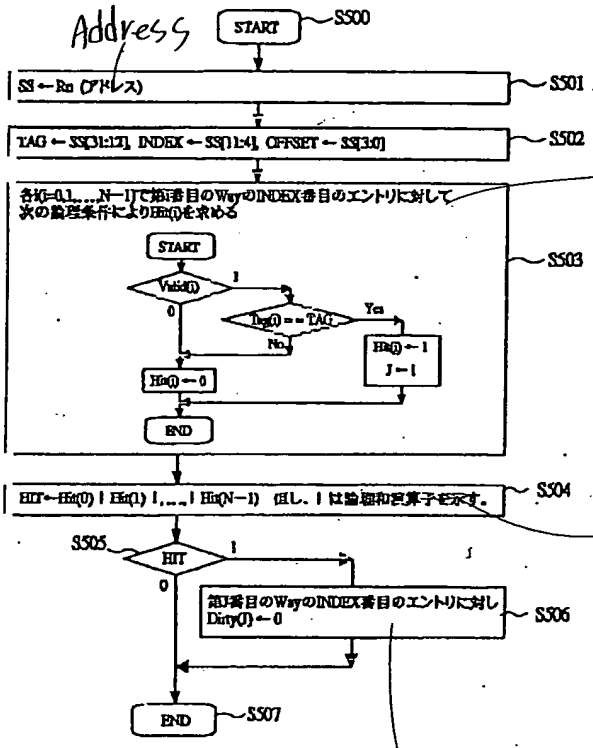
! shows the logical OR operator

Dirty(J) to the INDEXth entry of the Jth Way

Figure 9

【図9】

図 9



Hit(i) is caluculated fo
the INDEXth entry of t
ith Way of each i (i= 0,
N-1) according to the
following logic requirm

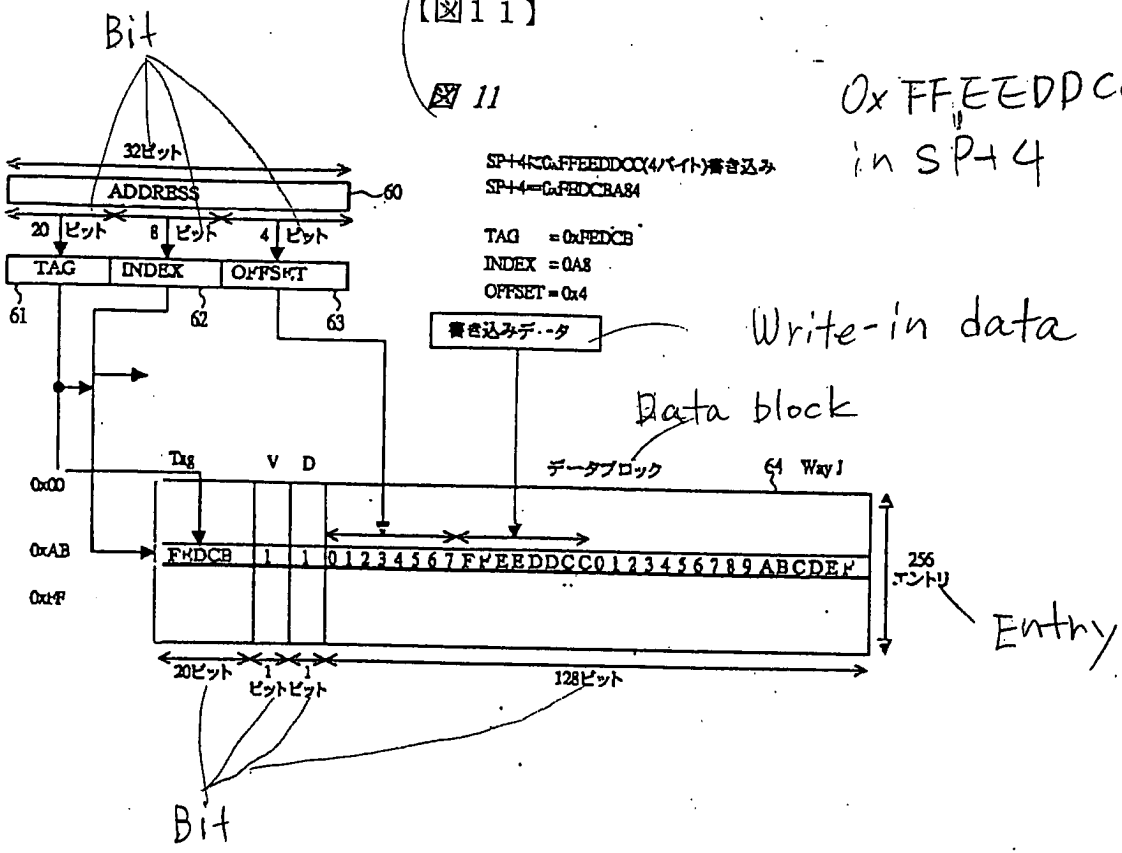
I shows the logical OR
operator

Dirty(J) to the INDEXth entry of
the Jth Way

Figure11

【図11】

図 11

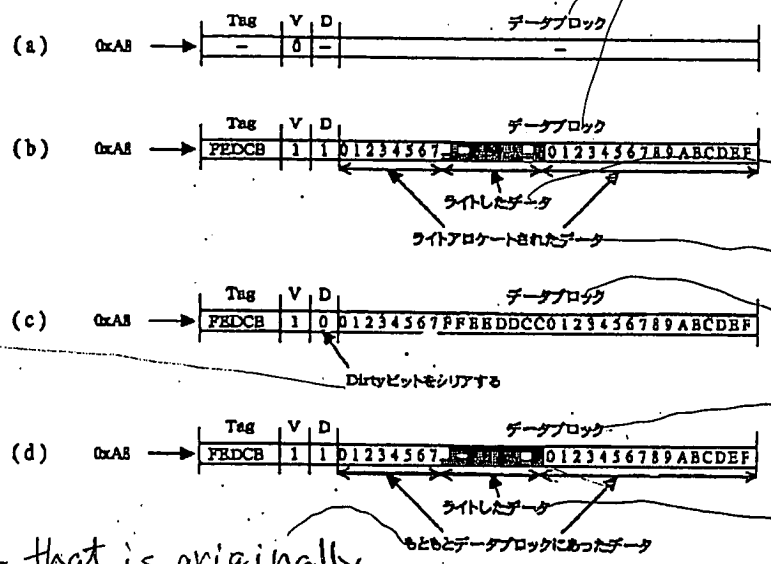


0x FFEEDDCC (4 bytes) is written in SP+4

Figure 12

【図12】

図 12



Data block

Write-in data

Write-allocate data

Data block

Dirty bit is cleared

Data block

Write-in data

Data that is originally present in the data block.